REMARKS/ARGUMENTS

Amendments were made to the specification to specify the application numbers of the cited related applications. No new matter has been added.

Claims 1 and 4-8 are pending in the present application. Claims 2, 3, and 9-24 are canceled. Reconsideration of the claims is respectfully requested.

Applicants have amended some claims and canceled others. Applicants do not concede that the subject matter encompassed by the earlier presented claims is not patentable over the art cited by the Examiner. Applicants canceled and amended claims in this response solely to facilitate expeditious prosecution of this application. Applicants traverse all rejections and respectfully reserve the right to pursue the earlier-presented claims, and additional claims, in one or more continuing applications.

I. 35 U.S.C. § 101, Claims 9-24

The Examiner rejects claim 11 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. Applicant has canceled claims 9-24. Therefore, this rejection is moot.

II. 35 U.S.C. § 103, Obviousness,

The Examiner rejects claims 1, 4-8, 9, 12-16, 17, and 20-24 under 35 U.S.C. § 103 as obvious over *Alexander III et al.* (U.S. Patent No. 6,338,159) (hereinafter "*Alexander*") in view of document *JaViz: A client/server Java profiling tool* published in 2000 by *Kazi et al.* (hereinafter "*Kazi*"). This rejection is respectfully traversed. In regards to claim1, the Examiner asserts the following:

> Alexander(1) discloses:
>
> 1. A method, in a data processing system, for generating a minimized call tree data structure from trace data obtained from a plurality of executions of a computer program, comprising:
>
> obtaining a plurality of call tree data structures (col. 2, lines 32-33 "a call stack ... one or more nodes in the tree") corresponding to the trace data (col. 2, lines 28-29 "trace information ... obtained...") for the plurality of executions of the computer program (col. 2, lines, 38-39 "...number of Java bytecodes executed in each method ... called").
>
> Alexander(1)does not disclose generating a minimized call tree data structure from the plurality of call tree data structures wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures; and outputting the minimized call tree data structure.
>
> However Kazi discloses in an analogous computer system generating a minimized call tree data structure from the plurality of call tree data structures (Kazi page 100 "Tree generation ... merged trace files to create an output file containing the dynamic execution tree") wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures (Kazi page 100 "Tree generation.. . Run-time

> statistics generation.. .Each detailed ... trace file is analyzed to gather the total number of calls made to each method, the maximum, minimum, and average execution times, and the standard deviation of the execution time for each method"). Thereby minimizing the display for execution.; and outputting the minimized call tree data structure (Kazi page 100 "Tree generation ... merged trace files to create an output file containing the dynamic execution tree").
>
> Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of generating a minimized call tree data structure from the plurality of call tree data structures wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures; and outputting the minimized call tree data structure as taught by Kazi into the method for providing the trace information as taught by Alexander(1). The modification would be obvious because of one of ordinary skill in the art would be motivated to generate a minimize call tree data structure from the trace data to provide a performance analysis tool to allow developer to determine the execution times have high of low variance as suggested by Kazi (page 115, "Conclusion").

Office Action dated December 7, 2007, pages 5-6.

The Examiner bears the burden of establishing a prima facie case of obviousness based on prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The prior art reference (or references when combined) must teach or suggest all the claim limitations. *In re Royka,* 490 F.2d 981, 180 USPQ 580 (CCPA 1974). In determining obviousness, the scope and content of the prior art are... determined; differences between the prior art and the claims at issue are... ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background the obviousness or non-obviousness of the subject matter is determined. *Graham v. John Deere Co.*, 383 U.S. 1 (1966). Often, it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art, all in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue. *KSR Int'l. Co. v. Teleflex, Inc.*, No. 04-1350 (U.S. Apr. 30, 2007).

II.A. <u>The Examiner has failed to state a *prima facie* case of obviousness because the references do not teach or suggest all the features of claim 1.</u>

II.A.i. Claim 1

Regarding claims 1 and 4, the Examiner has failed to state a *prima facie* obviousness rejection because the proposed combination of *Alexander* and *Kazi* does not teach or suggest all the features of the claims. Claim 1 is as follows:

> 1. A method, in a data processing system, for generating a minimized call tree data structure from trace data obtained from a plurality of executions of a computer program, comprising:
>
> obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program;
>
> generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures; and
>
> outputting the minimized call tree data structure.

The Examiner has failed to state a prima facie obviousness rejection against claim 1 because neither *Alexander* nor *Kazi* teach or suggest "generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures." The Examiner acknowledges that *Alexander* fails to teach or suggest this feature. However, the Examiner believes that *Kazi* teaches this feature. The Examiner cites:

> *Tree generation.* The tree generation step analyzes the merged trace files to create an output file containing the dynamic execution tree for a given client or server program...
>
> *Run-time statistics generation*...Each detailed .prf trace file is analyzed to gather the total number of calls made to each method, the maximum, minimum, and average execution times, and the standard deviation of the execution time for each method.

Kazi, pg 100.

The Examiner cites to this portion of *Kazi* as teaching the feature "generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures," as in claim 1. However, *Kazi*, when viewed as a whole, teaches a method that provides the ability to generate an execution tree that shows all the trace execution threads across Java Virtual Machine (JVM) boundaries, in a large-scale client/server environment using the Java remote method invocation (RMI) facility. Implementation of the method described in *Kazi*, requires merging three trace files: the detailed trace, the client profile, and the server profile. These files are merged to produce one detailed trace for each JVM. After the merge step is completed, a dynamic execution tree is produced that shows *every* RMI trace execution call recorded on each JVM.

Kazi fails to make up for *Alexender's* deficiencies because *Kazi* does not teach the feature "generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the

plurality of call tree data structures" as recited in claim 1. Instead, *Kazi* teaches the generation of an execution tree that shows *every* trace execution call recorded on each JVM.

Accordingly, because neither *Alexander* and *Kazi*, teach or suggest all the features of claim 1, the proposed combination of *Alexander* and *Kazi* when considered as a whole does not teach or suggest all the features of claim 1. Therefore, the Examiner fails to state a *prima facie* obviousness rejection of claim 1 or of any other claim.

II.A.ii. Claim 4

Claim 4 is as follows:

> 4. The method of claim 1, wherein generating the minimized call tree data structure includes:
>
> copying a first call tree data structure; and
>
> walking a second call tree data structure over the first call tree data structure to generate the minimized call tree data structure.

Regarding claim 4, the Examiner has failed to state a *prima facie* obviousness rejection because the reference does not teach or suggest all the features of claim 4. In rejecting claim 4 the Examiner incorporated the rejection of claim 1. The Examiner further asserted the obviousness rejection of claim 4, citing *Alexander* as follows:

> If it is an exit event, the tree is traversed to the parent (using the parent pointer), and the current tree node is set equal to the parent node (step **178**). At this point, the tree can be dynamically pruned in order to reduce the amount of memory dedicated to its maintenance (step **179**).

Alexander, col. 6, lines 27-31.

This cited portion of *Alexander* discloses a method of traversing, i.e. walking, a tree after first determining whether the trace event is an *exit* event. *Alexande*r further teaches that the tree is pruned only where the trace event has been identified as an exit event. However, no teaching or suggestion is made in *Alexander* of generating a minimized call tree data structure as recited in claim 4. For this reason, *Alexander* fails to teach or suggest all the features of claim 4. Accordingly, the Examiner fails to state a *prima facie* obviousness rejection of claim 4.

Furthermore, because the rejection of claim 4 incorporated the rejection of claim 1, the same distinctions between the cited reference vis-à-vis claim 1 applies to claim 4. Consequently, because the remaining claims depend from claims 1 or 4, the proposed combination of *Alexander* and *Kaz*i when considered as a whole does not teach of suggest all the features of the dependent claims. Thus, under the standards of *In re Royka*, the Examiner fails to state a *prima facie* obviousness rejection of claims 1 and 4-8. Therefore, the rejection of claims 1 and 4-8 under 35 U.S.C. §103 has been overcome.

III. Conclusion

The subject application is patentable over the cited references and should now be in condition for allowance. The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: March 12, 2008

Respectfully submitted,

/Theodore D. Fay, III/

Theodore D. Fay, III
Reg. No. 48,504
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants

TF/lj-m